INCREASING INTEROPERABILITY BY CONVERGING SERVICES IN CONSTRAINT-BASED-ROUTING NETWORKS

Dag-Anders Brunstad, James B. Michael
Computer Science Department, Naval Postgraduate School
Code CS/Mj, 833 Dryer Road, Monterey 93943-5118
Tel: +1 (831) 656-2655, Fax: +1 (831) 656-2814
E-mail: dag.brunstad@ftd.mil.no / bmichael@nps.navy.mil

June 1, 2000

Abstract: *Putting the customer requirements in the center and designing new systems around them is considered the right way to design information systems. In this article we claim that requirements change so fast in the information age that customer needs are unknown and impossible to correctly predict for both the customer and the system designer. Basing the design on a model that can support changes by ensuring flexibility and interoperability must therefore be given more attention. We introduce an object-based network interoperability model, based on the TCP/IP hourglass principle, where every system should be designed as a network object. We also suggest a set of constraints on object behavior with respect to interoperability. By using the evolving constraint-based routing technology, we can integrate all networks into one global system that both supports all services and ensures fully communication-interoperable systems.*

KEYWORDS: Interoperability, Constraint-based routing, Networks, System architecture

1. Introduction

After three to four decades of continuous improvement of the performance of information systems with interoperability treated as a secondary issue, we have ended up with myriads of non-interoperable (stovepipe) systems. The defense community, both in U.S. and elsewhere have expended considerable resources to make these systems meet application-specific demands for security and reliability. Several of these systems are still working well as individual systems and are too valuable to be replaced in a short timeframe. In contrast, as a rule of thumb, in the commercial sector a computer system becomes obsolete after 18 months. Another important factor that changes the way computerized systems are built is the shift in user market. The arrival of the information age has led to an explosion of

distributed users, databases, and communication networks in the commercial sector. Twenty years ago the government and military were dominant forces in steering the development of information systems. Today, when almost every company in the commercial sector uses several kinds of information systems, government and military systems constitute less than ten percent of the market. This has lessened the interest shown by the private sector for providing custom information systems and enabling technology for this relatively small community. In addition, the willingness to pay the increasing cost of specialized government and military systems is steadily decreasing.

One of the most important requirements for defense systems developed in the future will be interoperability. General Collin L. Powell described the demands for information systems like this “Give the battle field commander access to all the information he needs to win the war. And give it to him when he wants it and how he wants it.” The future expectations of computer and communication systems seems to be that anyone, anywhere, at anytime should be able to communicate with any other system. The question is how do we get there?

2. Change in the way systems are designed

2.1 Current Paradigm

The development process should start with the specification of an architecture, which serves as a roadmap for planning and performing development tasks. Just like a blueprint for a building, the network architecture is needed to give an overall view of the components of the system infrastructure and show how technology, users, processes, and tools fit together. The architecture also defines key strategies and objectives in addition to the network structure and the standards and methods applicable to the architecture. It is very common to divide up the tasks of complete design into a five-phase process after a project is selected for development [1]. The steps are: project initialization, planning and analysis, logical and physical design, implementation and testing, operation and maintenance. Whether a system is design bay following waterfall (the fore steps are done sequentially one time) or an evolutionary (the for steps are performed repeatedly until an acceptable solution is achieved) process model, most of these tasks are common in the design of new information systems. The goal is to cover most of the needs identified for the initial users and processes in a system, but in many cases system designers miss the global view of an information system as being part of a larger system.

2.2 New paradigm: Bringing the network to the center

One of the problems we see is the lack of focus on requirements for future interoperability with other information systems. Systems are in general designed only to take care of the current and the instantaneous foreseeable needs in a designated problem area. Let us take a look at some of the emerging management philosophies that the information systems that is design is going to support. “We’re not smart enough to predict the future, so we have to get better at reacting to it more quickly” has become the company slogan in General Electric. Jack Welch, the company’s widely quoted CEO, has earned respect as the golden child in management circles for implementation of “philosophies” that have transformed GE from a doomed old-fashion company, to a highly successful and customer responsive giant. Haeckel [2] argues that successful companies have developed a more rapid way of responding to the customer’s needs. He says “To understand as early as possible the customer’s underlying, unarticulated request, organizations must invest in collecting signals that may not appear to be a request at all.” The art of listening for implicit as well as explicit requests and to be able to comply rapidly too both is his recipe to success.

Economist’s like Davenport and Prusak [3] emphasize the problem of information overflow. In order to take care of the information in a company and bring it to a higher more useful and valuable stage, the information must be combined and transformed into knowledge. The authors define three levels of maturity for what systems display to the users: data (ones and zeros), information (processed and organized data made presentable to the user), and knowledge (information processed and organized into a higher level). System designers have developed techniques to organize data into information. In today’s world of information overflow, a higher level of organization of the available information is needed. Sources of information at all levels should be able to exchange information and enable the aggregation of knowledge from these sources.

Some of the experiences and theories we have discussed so far tell us that:

- The requirements of customers change at unpredictable rates.
- Neither the system designer nor the customer that the system is designed for can predict with certainty the system requirements that the near future might present.
- The technology in the system one designs might be obsolete before the design is complete.
- Current information is one of the most important assets of our time.
- Success depends on quick decisions which in turn depend on current and sorted information.
- It is necessary to try to anticipate and respond to the customer’s requirements before the customer knows about the requirements.

What type of system design is this leading us to? This trend of requirements seems to indicate that we need highly interoperable systems that can share and combine information and that are very flexible and easily changeable.

We claim that the focus on placing the customer in the center and creating the system according to his needs has been misinterpreted by system designers to mean deliver only part of the needed product. According to the business indicators we just looked at, neither designers nor customers know the real requirements for the system. The solution to this dilemma, selected by most designers, is to create a system after best guess and customize the system after the current and predictable needs of the customer. In our opinion this design strategy is the origin of the stovepipe systems that we see today. These system, are often working well for the purpose they where created for but are not properly designed to share information with or access information in other systems when the customers requirements change.

Most system designers seem to be solving the problem of unknown requirements for future use of the system in a similar way to what Isaac Newton did. To be able to explain our world as concise transactions in a mathematical way, he chose to overlook the creation of new systems and totally abounded Johannes Kepler's (1571 – 1630) theories about the harmonies in the overall system. Newton left the creation of new systems as the domain of God (the story of the Creation from one of the books of Pentateuch).

In order to create an information system that supports the processes of the ever-changing business environment might be a difficult task, but we have to wait for a while if we leave it to the powers beyond our control. The interoperability dilemma has been researched for a number of years and for systems that includes both hardware and software components, no other solution than to design a common reference for all systems have been found. If we manage to define such a reference, and design all our systems around it, we can look at it as the Keplerian view of harmonic systems. The common base can enable creation of new systems inside the overall system because all units are interoperable. To get away from stovepipe systems and having to spend valuable resources to make the systems capable of exchanging information, at a later point because requirements have changed, must be the focus of the development process in the future.

We do not argue the importance for the individual system designers to put the customer in focus. The importance of this is undisputable, but we need to give the system designer another frame of reference. The designer must have a defined interface towards the overall system that the design is going to be a part of to make

sure that the system can exchange data, information, or knowledge with any other system.

Figure 1 shows an example of ho systems traditionally have been designed and connected together.

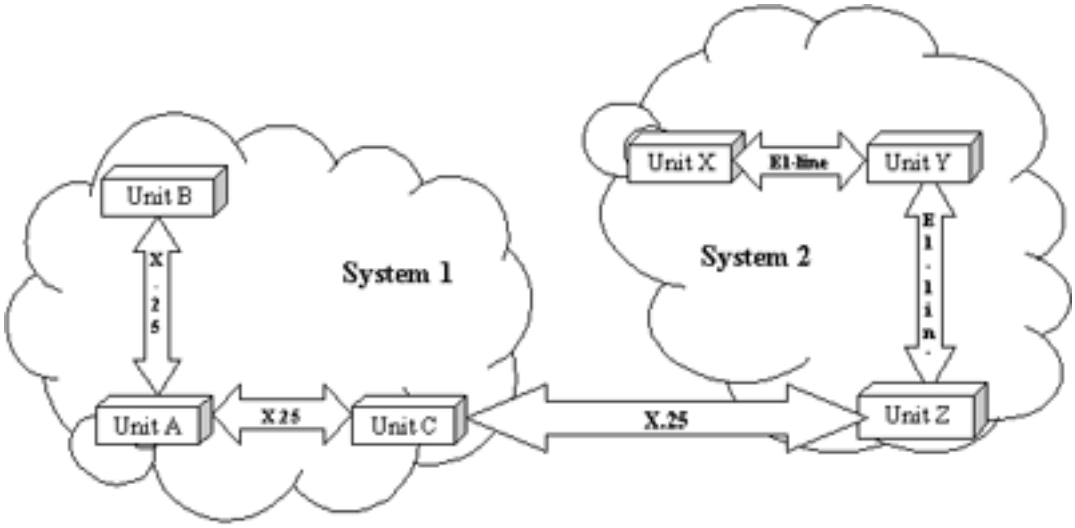

Figure 1. Stovepipe systems, the traditional system design

In this example we show two systems that had to exchange information after they where individually designed without consideration of interoperability. In some cases the system designers have thought ahead and developed an interface that will make the system able to communicate with other systems. Numerous of attempts through the years on standardization on one type of interface have failed. As a result, it has been considered the best possible design to leave the current "standard" interface as a communication gateway to the system.

We claim that we need to change this way of thinking of systems to achieve good interoperable system design. With the development process that we referred to as current practice, we can recognize a number of design weaknesses:

- Systems may not be designed for ease of data exchange with all other systems. A communication interface might be prepared on individual systems, but there is no common agreement to ensure that it will match other systems.

- The design of systems individually without making them a part of a whole requires individual administration and control of each system. When the systems are combined, administration and control of systems may not scale well in terms of global control.

- Connection-oriented communication is very resource demanding and often blocks more resources than it effectively uses.

- System survivability is difficult to obtain without duplicating recourses if communication between systems is implemented on dedicated systems. In figure 1 ordinary connection-oriented communication and end-to-end communication lines are used and additional resources have to be used to ensure availability.

- Up scaling of the system is difficult because the system is not designed for infinite scalability.

The list above is not comprehensive, but illustrates some of the most important interoperability shortcomings of the current methods of design.

3. Converging networks

3.1The hourglass model

A well-known system design that has managed to extend the life cycle for a number of information systems and handle increasing demand for scalability is the Internet. One of the objectives for the researchers who designed the original Internet was to make it possible to incorporate new technologies without discarding existing networks. Because of this objective we can say that the Internet as originally designed to connect stovepipe systems. The main reason for the success of the Internet can be credited to the IP "hourglass" model; the IP protocol has as a stable point of reference provided a consistent, best-effort service interface that has permitted the relatively independent development of applications and underlying networking technology.

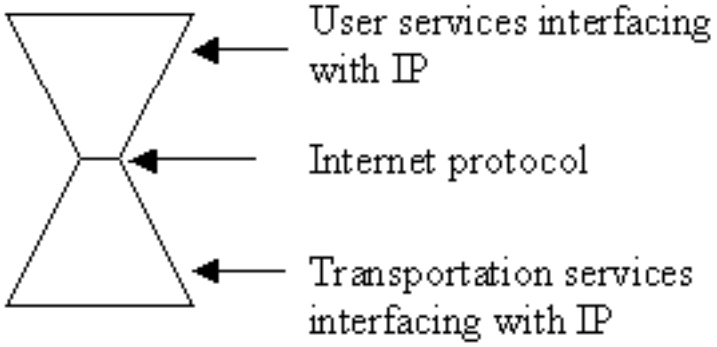

Figure 2 The IP hourglass

We are currently witnessing that increased research on network services that will enable IP networks to offer new types of services. Constraint-based-routing offers a way to implement policy and quality-of-service distinctions needed by different applications.

Together with additional research on multicasting techniques, that decrease the bandwidth demand dramatically, this has convinced most of the network society that it in the near future it will be possible to integrate all types of network services into one network. A network like this must be able to accommodate requirements for all types of communication whether it is voice, video, or data. In addition, network routing policy and enforcement of the routing quality-of-service policy must be implemented in an overall network management system.

3.2 Increasing interoperability with an object-networking model

With the hourglass-model in mind let us try to illustrate a model for an all-purpose communication network with the transportation services in center.

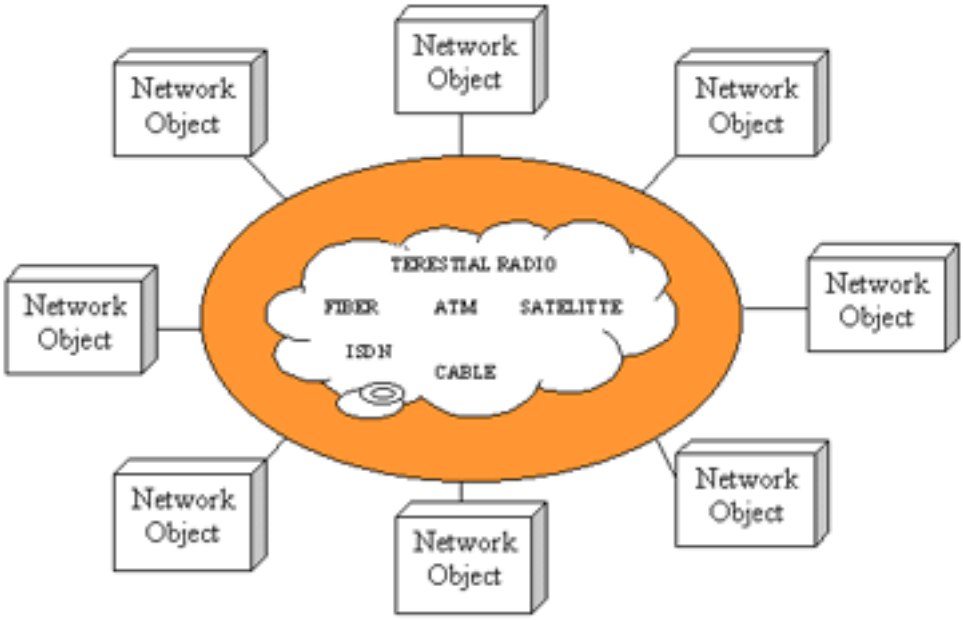

Figure 3 Object networking model, where transportation services are hidden to the network objects

As we can see from figure 3, the model of object networking builds on the principles of the Internet hour-glass model. The IP layer of the protocol stack implements the fixed level of reference that is needed to ensure interoperability. As new systems are created, they are added to the network as “just another” network participant. We will return to the network participant later and define a set of rules that it must comply with to be a well-behaved network object.

Redefining the focus of what is in the center of our system gives the designer of a new system a fixed reference point and the designer knows what interface to deal

with when the system is going to be networked. Compared to the development philosophy of dealing with connectivity to other systems as they show up, this approach to networking can eliminate the previously listed interoperability problems. When designing a good network object for this architecture, the object should have an IP interface and be able to share information and controls with the rest of the objects in the network. To adopt the object-oriented networking model, as a comprehensive way of supporting communication for voice, video, and data, two major tasks must be accomplished:

- Implement in IP networks both technology for routing of traffic that requires different levels of services and multicast techniques.

- Choose and adopt the best reference point in the stack of protocols, which as of today still is IP. This choice can change in the future, but with the size and the number of applications the Internet has taken on, we know that technology for stage-deployed implementation will be available when it is necessary to replace IP.

4. Interoperability enabling technologies

The best-effort service has worked fine for most traditional Internet applications (such as FTP and email), but it is interconnected with newly emerging real-time, multimedia applications such as Internet telephony, video-conferencing, and video-on-demand. These applications require the stable quality and delivery rates that we normally get through connection-oriented services. In other words, if we want to use IP-based networks for these new applications, we require better transmission services than best effort.

For several years extensive research has been conducted to implement routing and connection features that can distinguish between the different levels of service that are needed for a specific network communication session in IP networks. The implementation of the research results has been slowed down because the commercial sector has tended to throw increased bandwidth at the problem. The focus on wireless networks has again set the focus on the need to come up with bandwidth-economical solutions. In the RF-sector, bandwidth is a very scarce resource, and the demand for solutions with less waste of bandwidth is never ending. We now see the evolution in different kinds of network architectures towards more flexible support for multiple service categories.

QoS-based routing is defined in [RFC2386] as “A routing mechanism under which paths for flows are determined based on some knowledge of resource availability

in the network as well as the QoS requirement of the flows." To keep our concepts clear, we need to explain two relevant concepts called policy-based routing and constraint-based routing. Policy-based routing indicates that the routing decision is not based on the knowledge of the network topology and metrics, but on administrative policies. A policy may, for instance, prohibit a traffic flow from using a specific link for security reasons, independent of capacity and quality issues. Network management operators usually statically configure policy-based routing.

In contrast constraint-based routing refers to computing routes that are subject to multiple constraints. These constraints can include QoS constraints (e.g.,delay, jitter, bandwidth) and policy constraints (e.g., insecure routes) For this reason both QoS-based routing and policy-based routing can be considered special cases of constraint-based routing.

QoS-routing according to the DiffServ model [RFC 2475] currently looks to be the most promising model available. Instead of maintaining individual flows on all routers (like in the IntServ model [RFC 1633]), flows are divided into different types of classes, and packets that come to a server receive a service based on its labeled importance. In IPv4 the type of service field (renamed to Differentiated Service, DS) is used as label for what quality the packet requires. A routing policy must be maintained in each router to enable interpretation of and decision making for that router based on the DS label. When the data enters the routing network, its DS-class is identified. The IP-datagram is marked with the service it belongs to and sent on its way through the network. Routers in the path look in the IP header to determine where to send the packet just like in best-effort routing. But, in addition, the router checks what service class the packet belongs to. Packets with different service classes have different queues in the routers, and in this way the router can distinguish between packets that need higher rather than lower priority. The idea is that a packet containing an email can wait to be routed for a few seconds while priority is given to a packet containing real-time voice data for a telephone conversation.

Implementation of the DiffServ model requires several new routing functions. First of all, admission control must be implemented. The network must have the ability to refuse to take on more customers when the demand exceeds a certain capacity. Second, the routers must have a feature for packet scheduling. As mentioned when going through the model of how Diffserve works, the routers must have a method to threat different classes differently (e.g., different queues). Third, a scheme for traffic classification must be developed. This means that the network must sort the network traffic into different flows or classes based on the need for service. Last

but not least, functions for implementation of policies to allocate the network resources must also be implemented.

Together with public key infrastructure and multicasting solutions (beyond the scope of this paper), the suggested models for QoS routing brings us close to the goal of serving all types of traffic in one network [4]. As mentioned earlier, the DiffServ model looks most promising for several reasons. First of all the IntServ model puts several extra processing and storage tasks on all the routers in the network, which introduces concerns about the use of resources. In addition, the IntServ model has several security problems that can result in implementation difficulties because more traffic flow information is passed between routers.

5. Requirements for the network object

5.1A sequence of sense-decide-act modules

To ensure a good design in our object networking architecture and to make sure that the network objects of the system will be interoperable, several requirements must be established for them. We can divide the behavior of any information system into a series of modules that sense, decide, or act. We believe that a well performing information system consists of a series of sense-decide-act sequences, comparable to John R. Boyd's OODA loop [5] that characterizes a well performing decision maker. Ultimately any function of a system will perform one of these functions to contribute to the overall system task. The most flexible design that makes changes and reuse of network objects possible is to make an individual network object for each of the sense–decide–act modules. If this is done properly, any network object can be addressed individually and the module can be reused, as a part of any system. In cases where a complete division into sense-decide-act functions is impractical, one should aim at designing complete sense-decide-act sequences. A system performing incomplete sense-decide-act sequences is normally an indicator of less preferable design (e.g., sense-decide-decide-act will under normal circumstances work better as sense-decide-act-sense-decide-act).

We can identify a number of advantages in this system development process. First, it is very easy to update and replace pieces of the system as they become obsolete. Second, media interoperability becomes natural in system architecture like this. End systems can easily be share between information systems. As an example, we can imagine a sensor that gets used by several systems. If the data- and semantic-interoperability problems are solved, then we can reuse sensor and decision modules to address changing requirements. For defense systems, can this be an advantage. Even if the trend is to use more and more COTS products, specialized

systems need to be developed to support special defense requirements. These specialized systems tend to have longer life cycles than other systems and can be reused as part of a modernized system. A third advantage is a more natural support for evolutionary design. New solutions for parts of the system can more easily be implemented as new technology becomes available or user requirements change. Yet another advantage is the possibility for several users to share the cost and complexity of building and maintaining a system, which can bring it inside the range of affordability of new user groups.

5.2 The right interface for a network object

In order to ensure sufficient communication interoperability between the network objects and the rest of the network, the object must have a standardized interface. We have already motivated the fact that the system must interface to the routable-networks interface. IP-based networks supported by constraint-based routing services are the current routable type of interface that can support communication interoperability for different (ultimately all) systems.

In order to support the connection of legacy systems, the network object interface must support some way of piping data between such systems over the network. For example, the MIME protocol supports such an encapsulation interface.

If the architecture should be able to support all types of networks objects, then the importance of security solutions is going to grow with support for security down to the individual application object in each network object. When the communication function has an open global structure, every information exchange must be authenticated to avoid the possibility of creating chaos. In addition, a lot of information systems will require confidentiality. Some of the underlying security issues are solvable using PKI mechanisms, identification and authentication. A scalable solution for PKI infrastructures is yet to be developed, but this does not prohibit implementations where information systems distribute their own keys for the affected domain.

The main purpose of designing a network where the different systems use a common communication structure is to be able to control and share data remotely. One available tool for this is a request-reply protocol like Simple Network Management Protocol (SNMP) administrated from a Management Information Base (MIB-II).

We can sum up five identified basic functions the network-object interface must support:

1. It must have a routable communication interface function. A good example is the standard LAN interface (100 Base-T of FDDI) used in most of today's data communication networks.
2. An interface function for encapsulating data from systems that was not initially designed as a network object.
3. A security interface to support security down to individual application object level. A public key infrastructure type of interface function will solve this problem.
4. Quality-of-Service interface function. The information package sent from the object must be labeled with necessary routing requirements information so that a constraint based routing system can support it with proper routing service.
5. A management interface function to enable the network object to send or receive controls.

6. Summary

In this paper we have suggested a change in the way overall communication architecture between systems is designed. The rapid changes in information system functionality and the need to reuse parts of an information system force us to think of all information systems as part of a whole. New technology like constraint-based routing, multicast routing, and pubic key infrastructure makes it feasible to integrate all communication services into one communication network. This will require a change in thinking from the way we design information systems today. We cannot expect individual vendors to cooperate with every competitor and deliver solutions that solve the communication interoperability problem. Government as well as commercial companies will have to make a choice of a common platform to build their systems on to ensure interoperability among systems. The trends in communication show us that because of the widespread application of Internet solutions, negative lock-in effects from building this platform with IP as a fixed reference is very small. When it is relevant to implement technology to replace IP, it will come with solutions for stage-deployed implementation.

REFERENCES

[1] J. A Hoffer, J. F. George, J. S. Valacich, Modern System Analysis & Design, Addison-Wesley, 1998

[2] S. H. Haeckel: Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations", Harvard Business School Press, 1999.

[3] T. H. Davenport, L. Prusak: Working Knowledge: How Organizations Manage What they Know, Harvard Business School Press, 1998.

[4] Shigang Chen, "Routing Support for Providing Guaranteed End-to-End Quality-of-Service", Ph.D. thesis, University of Illinois Urban-Champaign, May 1999, http://cairo.cs.uiuc.edu/papers/Scthesis.ps

[5] John R. Boyd, "A Discourse on Winning and Losing," unpublished briefing and essays, Air University Library, document no. MU 43947 (August 1987).

[RFC 2386] "A Framework for QoS-based Routing in the Internet", IETF

[RFC 1633] "Integrated Services in the Internet Architecture", IETF

[RFC 2475] "An Architecture for Differentiated Services", IETF